Report

The design decisions I made, for example the main While loop in “runOneSimulation”, were based off the most efficient answer I could come up with, and what would take the least amount of time for the computer. I initially thought a couple of nested for loops in a while loop was the best choice, but I quickly realized this made the least sense to do, as it was many unnecessary loops and there’s no way to guess the number of times it would have to iterate, so the results wouldn’t be precise either. I assumed that there would always be at least one car passing through the protected section because if the left light or right light is green then the queues should be dequeuing. I also decided to add a couple other functions to help with the Queues and the probability aspect of the vehicles entering the queues. I added an “initialize” function which initializes the queues and setting the front, rear, and queue count to 0 or NULL. This is more efficient and makes the code a lot cleaner as it is easier to read and less repetition. I have also added a probability function for the same reasons as I would’ve had to repeat the same code multiple times, making it harder to read.

I have also assumed that the lights will change then the vehicles would queue so I decided to use two integers that represent a red light and green light. Another assumption was that the vehicles would dequeue if that light was green and it could still queue in the same direction as it would immediately dequeue if the light was green.

A couple experiments I performed were when the vehicles were arriving at the same rate in both directions, and the vehicles arriving is higher in one direction than the other.
For the first experiment, I used 0.5 as the rate the vehicles were arriving for both directions, and 4 as the time, also for both directions as the time periods would usually be the same for both directions. I noticed that this gave similar results for both directions as shown in the below code snippet:

```
$ ./runSimulations 0.5 0.5 4 4

Results (averaged over 100 runs):
        From left:
                number of vehicles:     251.080000
                average waiting time:   9.320000
                maximum waiting time:   4.000000
                clearance time:         8.450000
        From right:
                number of vehicles:     247.330000
                average waiting time:   8.490000
                maximum waiting time:   4.000000
                clearance time:         9.280000
```

I then conducted another experiment in which the time periods were different. This showed quite a big difference in the results from both directions. This showed that a lower time period would result in a higher amount of cars left in the queues and a lower waiting time. The results are shown in the below code snippet:

```
$ ./runSimulations 0.5 0.5 3 6

Results (averaged over 100 runs):
      From left:
            number of vehicles:     251.500000
            average waiting time:   0.290000
            maximum waiting time:   6.000000
            clearance time:         85.170000
      From right:
            number of vehicles:     252.040000
            average waiting time:   85.200000
            maximum waiting time:   3.000000
```

The final experiment I conducted was where all the command-line parameters were different. This revealed that the number of vehicles was much greater from the left direction as there was a lower time, even though the rate of vehicles was significantly lower than in the right direction.

```
$ ./runSimulations 0.3 0.7 4 6

Results (averaged over 100 runs):
      From left:
            number of vehicles:     348.920000
            average waiting time:   0.070000
            maximum waiting time:   6.000000
            clearance time:         150.150000
      From right:
            number of vehicles:     150.220000
            average waiting time:   150.190000
            maximum waiting time:   4.000000
            clearance time:         0.010000
```

This is an example of output from one run of the code:

```
bash-3.2$ ./runSimulations .4 .5 3 2

Parameter values:
        From left:
                traffic arrival rate:   0.400000
                traffic light period:   3.000000
        From right:
                traffic arrival rate:   0.500000
                traffic light period:   2.000000
Results (averaged over 100 runs):
        From left:
                number of vehicles:     299.150000
                average waiting time:   50.450000
                maximum waiting time:   2.000000
                clearance time:         9.250000
        From right:
                number of vehicles:     249.760000
                average waiting time:   9.280000
                maximum waiting time:   3.000000
                clearance time:         50.430000

bash-3.2$
```

I also interpreted the average waiting time as the time of vehicles + the time it takes to clear the vehicles. My code does work how I expected except for the maximum waiting time, as I initially thought the maximum waiting time would be for each vehicle, which would be the time for that direction.